Java™ Web Service Developer Pack JAXR API편(3)

Java 기술과 JAXR(1)

정리 · 최원석 | 한국 썬 시스템엔지니어링 본부 대리

지난 호까지는 Java™ Web Service Developer Pack을 본격적으로 설명하기에 앞서 웹 서비스에 대한 기본적인 개념과 관련 기술들에 대해 조명해보았다. 이를 모두 이해했다는 전제하에 이번 호에는 WSDP에서 제공하는 Java 기술과 툴을 소개하고, 그중 JAXR API의 클라이언트 및 제공자와 패키지, 나아가 JAXR API를 이용한 예제 등에 대해 집중적으로 살펴본다.

■ 웹 서비스를 위한 Java 기술과 툴

웹 서비스의 내용, 즉 비즈니스 로직은 어떠한 프로그래밍 언어로도 작성할 수 있다. 그러나 이식성과 같은 애플리케이션 개발에 있어서 이상적인 조건들을 고려하면 Java 프로그래밍 언어가 웹 서비스를 구축하고 웹 서비스를 사용하는 애플리케이션을 개발하는 데에 최적의 언어로 떠오를 수밖에 없다.

Java WSDP는 Java 프로그래밍 언어를 사용해 웹 서비스를 구축하고 사용하는 데에 필요한 Java API와 툴을 제공한다. 이들 API와 툴은 웹 서비스 제공자가 Java 언어를 사용해 웹 서비스를 표준 레지스트리에 등록하고, 서비스의 인터페이스를 기술하도록 해준다. 웹 서비스 요청자들은 API를 통해 Java 플랫폼 상에서 웹 서비스를 검색하고 사용할 수 있다. 웹 서비스 제공자와 요청자 모두 Java WSDP 패키지 상의 툴을 사용해 테스트해볼 수 있다. 또한 Java WSDP는 웹 애플리케이션(이 애플리케이션은 웹 서비스를 호출할 수도 있다)을 개발하기 위한 컴포넌트와 툴을 제공한다.

Java WSDP 배포판은 다음 웹 서비스 개발에 관련된 API와 툴을 제공한다. API들은 분리되어(즉 Java WSDP 툴과는 별도로 제공된다) Java XML Pack에서 제공되며, 각 API들은 분기마다 업데이트된다.

API들

- **Java API for XML Registries(JAXR) 1.0** : 이 API를 사용해 웹 서비스를 UDDI나 ebXML 레지스트리 등에 등록하거나 웹 서비스를 레지스트리 상에서 검색한다.
- **Java API for XML Messaging(JAXM) 1.0** : 이 API를 통해 XML 기반의 메시징 표준, 예를 들어 SOAP이나 ebXML 메시징에 부합되는 메시지를 전송한다. JAXR과 같은 API들은 JAXM을 이용해 메시지를 전송한다.

- **Java API for XML-based RPC(JAX-RPC) 1.0** : 이 API를 사용해 웹 서비스가 제공하는 조작들을 호출하는 XML 기반의 원격 프로시저 호출(RPC 호출)을 만든다.
- **Java API for XML Processing(JAXP) 1.2** : 이 API를 사용해 XML 문서를 처리한다. 예를 들어 웹 서비스 호출의 결과로 리턴된 XML 문서를 처리한다.

툴들

- **Java WSDP Registry Server 1.0** : 이는 테스트 레지스트리로 사용될 수 있는 UDDI 레지스트리를 구현한 것이다.
- **Tomcat Java Servlet & JavaServer Pages container 4.1-dev** : 이 툴은 Servlet과 JSP 페이지를 관리하고 실행할 수 있는 환경을 제공한다. 이것은 Java WSDP에 제공되어 테스트 서비스나 웹 애플리케이션을 만드는 데에 사용할 수 있다.
- **JavaServer Pages™(JSP™) Standard Tag Library(JSTL) 1.0** : 이 라이브러리는 자주 사용되는 JSP 동작들을 수행하는 데 사용될 수 있는 태그 집합을 제공한다. 예를 들면 리스트를 탐색하거나 조건이 부합될 때 처리할 동작들을 지정하는 데 사용할 수 있다. 이 표준 태그들은 Tomcat Java Servlet & JavaServer Pages를 포함한 다양한 JSP 컨테이너 상에서 사용될 수 있다.
- **Java Secure Socket Extension(JSSE) 1.0.2** : 이는 Java 패키지의 집합으로 안전한 인터넷 통신을 지원한다. SSL(Secure Socket Layer)과 TLS(Transport Layer Security) 프로토콜의 Java 버전과 데이터 암호화, 서버 인증, 메시지 무결성, 추가적인 클라이언트 인증 등을 지원한다. JSSE 1.0.2는 Java WSDP에 제공되어 SSL과 TSL을 사용하는 안전한 애플리케이션을 개발하는 데에 사용될 수 있다.
- **Ant Build Tool 1.4.1** : Ant는 Java 기반의 프로젝트를 만드는 데에 사용할 수 있는 툴이다. 이것은 웹 서비스 애플리케이션 등과 같은 XML 기반의 프로젝트를 만드는 데 매우 적합하다. 사실 Ant를 사용하면 프로젝트 빌드 동작을 XML 기반으로 설정할 수 있다.

■ JAXR

JAXR은 UDDI 또는 ebXML 등을 준수하는 표준 레지스트리에 접근하기 위한 Java API이다. 그러나 JAXR의 본래의 의도는 그것 이상이다. JAXR 스펙의 목표는 또 다른 미래의 레지스트리 표준을 지원하도록 하는 것이다. Java WSDP의 배포판의 JAXR API는 UDDI 레지스트리만을 지원한다. JAXR API를 사용해 ebXML 레지스트리를 지원하는 것은 오픈 소스 프로젝트에서 개발되고 있다. 자세한 사항은 OASIS ebXML 레지스트리 참조 구현 프로젝트(OASIS ebXML Registry Reference Implementation Project(ebxmlrr), http://ebxmlrr.sourceforge.net/)를 참조하기 바란다.

클라이언트와 제공자

JAXR API를 통해 요청을 보내거나 다루는 데에는 세 가지 역할이 존재한다. JAXR 클라이언트와 JAXR 제공자, 레지스트리 제공자가 그것이다. JAXR API는 JAXR 클라이언트와 제공자가 사용할 수 있는 인터페이스와 클래스들을 제공한다. JAXR 제공자는 JAXR API를 구현한다. 예를 들어 Java WSDP와 함께 제공되는 JAXR API의 참조 구현은 JAXR 제공자이다. 레지스트리 제공자는 레지스트리 스펙의 구현으로, 예를 들면 실제 UDDI 또는 ebXML 레지스트리이다.

JAXR 패키지들

JAXR API는 수많은 패키지들로 구성되어 있다. 그중 중요한 두 개의 패키지가 javax.xml.registry.infomodel과 javax.xml.registry이다.

· javax.xml.registry.infomodel

이 패키지는 JAXR 인포메이션 모델을 정의하는 인터페이스를 포함하고 있다. 인포메이션 모델은 어떠한 형태의 오브젝트가 레지스트리 내에 존재하며, 오브젝트의 타입이 어떻게 상호 관련되는지를 나타낸다.

JAXR 인포메이션 모델에서 인터페이스와 UDDI 스키마 내에서 데이터 구조와 같은 레지스트리 오브젝트 간의 명확한 대응을 눈여겨보기 바란다. 예를 들어 JAXR 인터페이스인 Organization은 UDDI의 businessEntity 데이터 구조에 명확하게 맵핑된다. 이와 유사하게 Service 인터페이스는 businessService 데이터 구조에 맵핑되며, ServiceBinding은 bindingTemplate에 맵핑된다.

이 패키지의 중요한 인터페이스들은 다음과 같다.

- **RegistryObject** : 이는 인포메이션 모델의 베이스 클래스이다. 이 클래스는 레지스트리 내 오브젝트에 대한 기본 정보를 제공한다. 또한 연관된 오브젝트에 접근할 수 있는 방법도 제공한다(예를 들면 Organization과 Service 오브젝트).
- **Organization** : 비즈니스 또는 다른 엔티티에 대한 정보를 제공하는 RegistryObject이다.
- **Service** : Organization 오브젝트에 의해 식별되는 비즈니스 또는 다른 엔티티가 제공하는 웹 서비스에 대한 정보를 제공하는 RegistryObject이다.
- **ServiceBinding** : Service 오브젝트에 의해 제공되는 특정 인터페이스에 접근하는 방법에 대한 정보를 포함하고 있는 RegistryObject이다. 예를 들

어 ServiceBinding은 웹 서비스가 제공하는 특정 조작이 HTTP 접속을 사용한 SOAP을 통해 접근하도록 지정할 수 있다. 동일한 Service 오브젝트에 대해서 다수의 ServiceBinding이 정의될 수 있다.

· javax.xml.registry

이 패키지는 JAXR 제공자를 통해서 레지스트리에 접근하기 위한 인터페이스와 클래스들을 포함하고 있다.

javax.xml.registry 패키지의 클래스와 인터페이스들은 UDDI API에 정의된 것들과 같은 레지스트리 메시지에 대해 대응하는 메쏘드들을 제공한다. 예를 들어 LifeCycleManager 인터페이스의 saveOrganization 메쏘드는 UDDI 퍼블리시 API 메시지인 save_business에 맵핑되며, BusinessQueryManager 인터페이스의 findOrganizations 메쏘드는 UDDI 질의 API 메시지인 find_business에 맵핑된다.

- **ConnectionFactory** : JAXR 접속을 위해 사용되는 팩토리 클래스들에 대한 추상 베이스 클래스이다. ConnectionFactory의 두 가지 중요한 메쏘드는 접속의 프로퍼티를 설정하는 데에 사용되는 setProperties와 접속을 생성하기 위해서 사용되는 createConnection이다.
- **Connection** : 이 클래스는 JAXR 클라이언트와 JAXR 제공자 간의 접속을 나타낸다.
- **RegistryService** : JAXR 제공자에 의해 구현되는 주요 인터페이스이다. 레지스트리 클라이언트는 이 인터페이스를 Connection 오브젝트를 통해 획득한다. RegistryService는 JAXR 클라이언트에게 JAXR 제공자의 기능에 따라서 제공되는 BusinessLifeCycleManager와 BusinessQueryManager와 같은 인터페이스를 획득할 수 있는 메쏘드를 제공한다.
- **LifeCycleManager** : 인포메이션 모델의 오브젝트에 의해 정의된 레지스트리 내의 정보에 대해 '라이프 사이클' 작업을 수행하기 위한 메쏘드를 제공하는 인터페이스다. 예를 들어 createOrganization 메쏘드는 Organization 오브젝트를 생성하며, createService 메쏘드는 Service 오브젝트를 생성한다.
- **BusinessLifeCycleManager** : LifeCycleManager 인터페이스의 하위 인터페이스다. UDDI 인터페이스와 유사한 형태의 라이프 사이클 관리를 요청하기 위한 방법을 제공한다. 예를 들어 saveOrganizations는 Organization 오브젝트의 컬렉션을 레지스트리에 저장하며, deleteServices 메쏘드는 Service 오브젝트의 컬렉션을 레지스트리로부터 제거한다.
- **BusinessQueryManager** : 레지스트리에 질의를 하기 위한 메쏘드를 제공한다. 예를 들어 findOrganizations 메쏘드는 메쏘드 호출에 지정한 검색 요건에 맞는 모든 organization들에 대한 레지스트리를 검색한다.

JAXR 예제

지금까지 논의한 인터페이스들과 클래스들의 실제 사용 예를 JAXR 예제를 통해 살펴보자.

이 예제에서는 BooksToGo라는 회사를 상정하고 JAXR을 사용해 웹 서비스를 등록한 뒤 BoomingBusiness.com이라는 회사가 JAXR을 사용해 해당 서비스를 검색하는 과정을 살펴보도록 한다. 웹 서비스 등록을 위해서 JAXR 1.0 참조 구현을 사용했다.

BooksToGo사는 다수의 서적 소매상을 운영한다. 소매상의 부족한 부분을 보충하기 위해서 동사는 웹 사이트를 통해 고객들이 서적을 온라인 주문할 수 있도록 했다. 최근 동사는 온라인 주문 기능을 웹 서비스로 사용할 수 있게끔 하기로 결정했다. 즉 서비스를 비즈니스 레지스트리를 사용해 공개하는 것이다. 동사는 Java 플랫폼 상에서 수많은 시스템과 온라인 서적 주문 등의 애플리케이션을 운용한다. 그 결과 Java WSDP에 있는 Java API를 웹 서비스를 구현하는 데 사용하기로 했으며, 특히 웹 서비스 등록에 Java를 활용하기로 했다. BooksToGo사는 Java WSDP를 사용해 온라인 서적 주문 서비스를 제공하기로 결정했다.

한편 BoomingBusiness.com사는 직원들에게 웹 사이트를 제공한다. 웹 사이트는 다양한 고용인 서비스를 제공하는 포털의 역할을 하고 있다. 예를 들면 인적 자원과 관련된 정보로 H.R 연락처 리스트에 대한 링크를 제공하고 있으며, 그밖에도 휴가 계획 작성 등의 기능을 제공한다. 동사는 이제 여기에 사용자들이 요청했던 서적 주문 기능 추가를 고려중이다. 더욱이 이러한 확장 기능을 웹 기반 솔루션으로 구현하기로 결정한다. 특히 동사는 사용자가 서비스 형태를 요청하면 애플리케이션 프로그램이 그러한 형태의 서비스에 대한 비즈니스 레지스트리를 동적으로 검색하는 방식으로 처리될 수 있도록 확장하려고 한다. 이러한 접근 방법을 사용하면 애플리케이션은 어떠한 종류의 서비스든간에 애플리케이션 차원에서 다양하게 지원할 수 있으며, 이에 따라 사용자들은 다수의 서비스를 자유롭게 사용할 수 있게 된다. BoomingBusiness.com사의 IT 관리자들은 이러한 기술이 플랫폼 독립적이어서 특정 프로토콜이나 사양에 얽매이지 않게 되기를 바란다. 따라서 동사는 Java 프로그래밍 언어와 XML, Java WSDP를 사용한 솔루션을 구축하기로 결정한다.

BooksToGo사, 온라인 서적 주문 웹 서비스 등록

BooksToGo사는 Java WSDP 패키지의 JAXR API를 사용해 온라인 서적 주문 서비스를 웹 서비스로 등록한다. 서비스는 UDDI 레지스트리에 등록된다. 동사의 온라인 서적 주문 웹 서비스 등록 절차는 다음과 같다.

단계 1 레지스트리 업데이트를 위한 인증

레지스트리는 통상 컨텐츠의 업데이트를 위해서 사용자가 적절한 인증 절차를 밟도록 요구한다. 사용자가 레지스트리 제공자로부터 사용자 ID와 비밀번호를 받은 다음 서비스 등록과 같은 작업을 위한 인증에 이를 입력한다(Java WSDP 레지스트리 서버는 UDDI 레지스트리에 대한 구현이며, 접근시 사용자 ID와 비밀번호를 요구하지 않는다).

단계 2 레지스트리 제공자에 접속

BooksToGo사의 애플리케이션 개발 스탭들은 Java 프로그램을 사용한 레지스트리 접근 및 업데이트 작업을 수행한다. 맨 처음 고려할 사항이 레지스트리에 접속하는 것이다.

- **접속 성립** : 레지스트리에 접근하기 위해서 JAXR 클라이언트는 레지스트리 제공자와의 접속을 성립시켜야 한다. 이를 위해 클라이언트는 ConnectionFactory 클래스의 createconnection 메쏘드를 호출한다. 각 레지스트리 제공자는 서로 다르게 설정된 하나 이상의 ConnectionFactory 클래스들을 제공한다. 레지스트리 제공자가 미리 설정된 ConnectionFactory를 제공할 경우에도, 이를 JNDI(Java Naming and Directory Interface™) 인터페이스 등을 사용해 네이밍 서비스에 등록해야 한다. 따라서 JAXR 클라이언트가 미리 설정된 ConnectionFactory를 찾는 과정은 JNDI를 통해 수행되는 것이 가장 좋은 방법이다.

  JAXR 1.0 참조 구현은 ConnectionFactory를 제공한다. JAXR 클라이언트가 이 참조 구현을 사용해 레지스트리에 접근하면 클라이언트는 ConnectionFactory 클래스의 newInstance 메쏘드를 사용해 인스턴스를 획득하게 된다. 예를 들면 다음과 같이 코드를 작성한다.

```
ConnectionFactory factory =
    ConnectionFactory.newInstance();
```

- **접속 프로퍼티 설정** : JAXR 클라이언트는 ConnectionFactory의 프로퍼티를 설정해야 한다. 이 프로퍼티들 중 몇몇은 JAXR의 표준이며, 다른 것들은 JAXR 제공자에 의해서 정의된다. 표준 프로퍼티들은 다음과 같다.

-- **javax.xml.registry.queryManagerURL** : 이는 레지스트리 제공자의 질의 관리자 서비스에 대한 URL을 지정한다. 레지스트리 질의를 위한 API들의 URL이다.

-- **javax.xml.registry.lifeCycleManagerURL** : 이는 레지스트리 제공자의 라이프 사이클 매니저 서비스에 대한 URL 문자열이다. 레지스트리 업데이트 API의 URL이다.

-- **javax.xml.registry.factoryClass** : 이는 레지스트리 특정 JAXR 제공자의 ConnectionFactory 클래스에 대한 완전한 클래스명이다. 레지스트리 특정 JAXR 제공자는 JAXR API를 레지스트리에 특화된 방식으로 구현하고, JAXR 플러그 가능한 제공자에게 이 JAXR API를 레지스트리와 독립적인 방식으로 플러그인하는 JAXR 제공자이다.

JAXR 클라이언트는 프로퍼티를 지정하기 위해 패러미터들을 ConnectionFactory의 setParameter 메쏘드를 통해서 넘긴다.

```
+ Properties props = new Properties();
 props.setProperty(
 " javax.xml.registry.queryManagerURL ",
            " ... ");
 props.setProperty(
 " javax.xml.registry.lifeCycleManagerURL ",
            " ... ");
 props.setProperty(
 " javax.xml.registry.factoryClass ",
            " ... ");
 factory.setProperties(props);
```

- **BooksToGo 접속** : BooksToGo사는 ConnectionFactory 오브젝트를 생성하고 다음과 같은 방식으로 프로퍼티를 지정한다. 참고로 다음 소스 코드의 URL은 보기 편하도록 여러 줄로 나눴다. 실제 프로그램 상에서는 다음 URL을 한 줄로 작성해야 한다).

```
import javax.xml.registry.*;
...

String queryURL =
  "http://www-3.ibm.com/services/uddi/
    testregistry/inquiryapi";
String publishURL =
  "https://www-3.ibm.com/services/
  uddi/testregistry/protect/publishapi";
...
ConnectionFactory factory =
      ConnectionFactory.newInstance();
Properties props = new Properties();
 props.setProperty(
 "javax.xml.registry.queryManagerURL",
    queryURL);
```

```
props.setProperty(
"javax.xml.registry.lifeCycleManagerURL",
    publishURL);
props.setProperty(
"javax.xml.registry.factoryClass",
   "com.sun.xml.registry.uddi.
          ConnectionFactoryImpl");
factory.setProperties(props);

connection = factory.createConnection();
```

단계 3 인증 정보 제공

앞서 설명한 바와 같이 레지스트리 제공자는 전형적으로 사용자가 적절한 인증 정보를 사용자 ID와 비밀번호 형태로 제공해야만 레지스트리의 내용을 업데이트할 수 있도록 한다. 특히 UDDI와 같은 레지스트리 스펙은 특정 요청에 대해서는 인증 정보를 제공할 것을 요구한다. 예를 들어 UDDI 스펙은 UDDI 퍼블리쉬 API에서 UDDI 레지스트리 인증을 요구한다. JAXR의 관점에서 이는 UDDI 레지스트리 제공자가 JAXR 제공자로부터의 요청을 통해 웹 서비스를 레지스트리에 등록하거나 레지스트리에서 제거하기 위해서 인증 정보를 필요로 한다는 의미이다.

- **사용자 신원 증명 지정** : JAXR은 인증 정보를 JAXR 제공자에게 제공하고, 이것이 연이어 레지스트리 제공자에게 제공되도록 하는 방법을 제공한다. JAXR에서 인증 정보는 다수의 사용자 신원 증명 형태로 넘긴다. 사용자 신원 증명의 개념은 J2SE™(Java 2 Platform, Standard Edition)의 JAAS(Java Authentication and Authorization Service)에서 이야기하는 것과 동일한 개념이다. 레지스트리 접근에서 사용자 신원 증명을 넘겨주기 위해서 JAXR 클라이언트는 먼저 PasswordAuthentication 오브젝트를 생성한다. PasswordAuthentication은 Java 2 Platform의 java.net 패키지에 위치하고 있다. 클라이언트는 PasswordAuthentication 오브젝트를 생성하고 사용자명(문자열 형태)과 비밀번호(문자 배열 형태)를 지정한다.

```
import java.util.*;

String userName = "...";
String password = "...";

PasswordAuthentication passwdAuth =
  new PasswordAuthentication(
            userName, password.toCharArray());
```

JAXR 클라이언트는 사용자 신원 증명을 지정하기 위해 Connection 오브젝트의 setCredentials 메쏘드를 사용한다. 사용자 신원 증명은 해시(Hash) 셋 형태로 지정된다.

```
Set credentials = new HashSet();
  credentials.add(passwdAuth);
  connection.setCredentials(credentials);

BooksToGo Provides Credentials: BooksToGo provides its credentials as follows:
```

- **BooksToGo사에 사용자 신원 증명을 제공** : BooksToGo사는 다음과 같은 형태로 사용자 신원 증명을 지정한다.

```
import java.util.*;
...
String userName = "btguser";
String password = "btgpwd";
...

// Constructing Authorization  Token
// 인증 토큰 생성
PasswordAuthentication passwdAuth =
  new PasswordAuthentication(userName,
            password.toCharArray());

Set credentials = new HashSet();
  credentials.add(passwdAuth);
  connection.setCredentials(credentials);
```

단계 4 서비스 등록

서비스를 등록하기 위해서 JAXR 클라이언트는 다음과 같은 절차를 따른다.

- **registryService의 획득** : JAXR에서 레지스트리에 대한 등록이나 질의와 같은 모든 동작들은 registryService 인터페이스를 통해 이뤄진다. 이것은 JAXR 클라이언트가 접속을 획득한 뒤에 해당 접속과 연관된 registryService 인터페이스를 획득해야 한다는 의미이다.
  다음에서 볼 수 있듯이 클라이언트는 getRegistryService 메쏘드를 사용해 registryService 인터페이스를 획득한다.

```
RegistryService rs = connection.getRegistryService();
```

- **LifeCycleManager 획득** : JAXR 클라이언트가 registryService 인터페이스를 획득한 뒤에, 이 인터페이스를 사용해 특정 레지스트리를 요청한다. JAXR에서 'Life Cycle Manager'라는 용어는 비즈니스를 등록하거나 웹 서비스 등록을 업데이트하는 활동을 설명하는 데 쓰인다. 즉 인증을 필요로 하는 동작들이다. JAXR 제공자는 LifeCycleManager 인터페이스를 구현하며, 이 인터페이스를 사용해 JAXR 클라이언트는 레지스트리에 새로운 오브젝트를 생성하는 등의 작업을 할 수 있다. LifeCycleManager 인터페이스는 일반화된 API를 제공한다. LifeCycleManager의 하위 인터페이스인 BusinessLifeCycleManager는 비즈니스에 보다 중점을 둔 인터페이스를 제공한다. 이 인터페이스들을 얻으려면, JAXR의 getLifeCycleManager와 getBusinessLifeCycleManager 메쏘드를 사용한다.

```
LifeCycleManager lifeCycleManager =
        rs.getLifeCycleManager();
BusinessLifeCycleManager
  businessLifeCycleManager=
        rs.getBusinessLifeCycleManager();
```

- **레지스트리에 저장할 오브젝트의 생성** : LifeCycleManager 인터페이스는 다양한 팩토리 메쏘드들을 가지고 있으며, 이 메쏘드들을 사용해 레지스트리에 서비스를 등록할 수 있다. 메쏘드들의 명칭은 createInterface 형태이고, Interface 위치에 JAXR 정보 모델의 인터페이스명을 기술한다. 메쏘드 명칭의 예는 다음과 같다.

createOrganization
createService
createUser
createPersonName
createTelephoneNumber
createEmailAddress
createClassification

앞서 설명한 바와 같이 Organization은 UDDI 레지스트리 내의 businessEntity 데이터 구조에 맵핑된다. 또한 businessEntity와 마찬가지로 다른 종류의 데이터, 예를 들면 제공하는 서비스나 연락처, 비즈니스 계층 분류 등의 정보를 갖고 있다. 이러한 관련 정보는 정보 모델의 오브젝트 내에 포함된다. createInterface 메쏘드들을 사용해 이러한 오브젝트들을 생성할 수 있다. 예를 들어 다음 코드는 Organization을 생성하고 설명을 첨부한다. createInternationalString 메쏘드를 잘 살펴보기 바란다. 이 메쏘드는 다양한 Locale에 맞게 국제화된 문자열을 생성하는 데 사용된다.

```
BusinessLifeCycleManager lifeCycleManager = null;
...

Organization org =
  lifeCycleManager.createOrganization("MyBusiness");
  InternationalString orgDesc =
    lifeCycleManager.createInternationalString(
          "An excellent business");
org.setDescription(orgDesc);
```

- **BooksToGo사 등록** : BooksToGo사는 다음 정보를 자신의 비즈니스와 온라인 서적 주문 서비스에 등록하고자 한다.

```
사업명 : BooksToGo
연락처 : 주요 연락처 - Ramesh Mandava
          전화번호 - (887)1111111
          이메일 주소 - ramesh.mandava@BooksToGo.com
계층 분류 방법(분류, 코드) : NAICS(Book Stores", 451211)
서비스 : 온라인 서적 주문
서비스 바인딩 : 상세 정보 - JAXRPC(SOAP/HTTP) 기반의 바인딩
                액세스 포인트 - www.BooksToGo.com:8080/books/
```

계층 분류를 생성하기 전에 먼저 해야할 일이 있다. BooksToGo사는 먼저 레지스트리에 질의해 적절한 분류 체계(정식 명칭은 taxonomy)를 찾는다(자세한 내용은 '분류 확인' 부분을 참조). 이 경우에 사용된 분류 방식은 NAICS(North American Industry Classification System)이다. JAXR은 QueryManager 인터페이스와 하위 인터페이스인 BusinessQueryManager를 제공해 레지스트리에 대한 질의를 지원한다. BusinessQueryManager의 메쏘드인 findClassificationSchemeByName은 패러미터로 넘겨진 패턴에 근거해 분류 방법들을 검색한다. BooksToGo사는 NAICS 패턴 ntisgov:naics에 관심을 가지고 있다.

■ 마치며

지금까지 웹 서비스를 위한 Java API 중 JAXR에 대해 살펴보았다. 아울러 이를 이용한 예제도 다뤄보았는데, 여기서는 서비스 등록에 관한 것만 다뤘다. 다음 호에는 JAXR을 사용해 등록된 서비스를 검색하는 과정에 대해 알아볼 것이다.